

Technical Note: Radiotherapy dose calculations using GEANT4 and the Amazon Elastic Compute Cloud

C M Poole,^{1, a)} I Cornelius, J V Trapp, and C M Langton
*Discipline of Physics, Faculty of Science and Technology
 Queensland University of Technology, Brisbane, Australia*

(Dated: 20 January 2013)

Cloud computing allows for vast computational resources to be leveraged quickly and easily in bursts as and when required. Using the Amazon Elastic Compute Cloud and the Amazon Simple Storage Solution, we describe a technique that allows for Monte Carlo radiotherapy dose calculations to be performed using GEANT4 and executed in the cloud. Simulation cost and completion time was evaluated as a function of instance count using compute instances acquired via bidding on the Elastic Compute Cloud spot market. Bidding for instances on the instance spot market was found to be 35-60% of the cost of on-demand instances of the same type. Using the technique, we demonstrate the potential usefulness of cloud computing as a solution for rapid Monte Carlo simulation for radiotherapy dose calculation.

Keywords: cloud computing, Monte Carlo, GEANT4, radiotherapy

I. INTRODUCTION

GEANT4 is a C++ toolkit for the simulation of particle transport through geometry, and is used widely in the field of high energy physics¹; adoption of GEANT4 for radiotherapy treatment verification however, is increasing¹⁻⁶. Flexible geometry definition and physics process customisation provides the user with a high level of control, and the opportunity to simulate a wide range of radiotherapy techniques including brachytherapy, hadrontherapy and intensity modulated radiotherapy⁷. Significant computational overhead prevents the routine use of these Monte Carlo techniques in the clinical setting, however the advent of cloud computing provides a low cost and easy to maintain alternative to the set-up of dedicated compute hardware⁸, something that may be of particular benefit to clinics in rural and regional areas and developing countries. Indeed, several authors have explored the usefulness of the cloud for Monte Carlo simulation⁹⁻¹¹, the most notable of which uses Fluka for proton beam dose calculations on the Amazon Elastic Compute Cloud (Amazon Web Services LLC, USA)⁸.

Amazon Web Services (AWS) provide organisations and individuals with the opportunity to leverage unused or under utilised Amazon network capacity for the purposes of scalable service provision such as high demand web-hosting with volatile loading conditions and scientific computation problems requiring significant compute or memory resources¹². Under the AWS umbrella there are a number of specific services providing distinct capability, the most relevant of which for this study are discussed. Amazon Elastic Compute Cloud (EC2) provides scalable compute through a number of predefined instances, where an instance is a virtual hardware device (or physical hardware device for select cases) with

a predefined compute capability; the full gamut of instance types is outlined in table I. Compute capability of a particular instance type is described using the EC2 compute unit, where one compute unit is the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor¹². At creation, any EC2 instance may have custom user data parsed to it, the user data itself may take on any form whether it be binary, ASCII or otherwise - subsequently this user data may be used to uniquely configure running tasks on the instances, or indeed the instance itself.

In addition to EC2, AWS provides a redundant storage solution for the persistence of data. Amazon Simple Storage Solution (S3) enables bulk upload and download of data associated with compute tasks, as well as provision for persisting the shutdown state of an instance - this is accomplished via the elastic block storage (EBS) virtual device which is backed by S3¹³. Access to the resources provided by EC2 and S3 can be performed programatically using the `boto` Python module¹⁴ or directly via the AWS dashboard using a web browser.

Access to most services associated with AWS attract a usage fee¹⁵. Charges associated with S3 are at fixed rates where storage volume and events such as disk input/output are charged separately. Three fee regimes are available for the user to select from when using the EC2 service. On demand usage attracts a flat hourly rate dependant to instance type, and a secondary fee structure provides the opportunity for substantially reduced hourly usage rates with the payment of a yearly subscription in order to reserve a dedicated instance. Dedicated instance reservation becomes increasingly economical as instance uptime and usage approaches 100%¹⁵. The third fee structure is delivered via a spot market where the user may enter the maximum bid price one is willing to pay for a given instance; price fluctuations of the spot market are governed by supply and demand on the market at the time. If the spot price exceeds the maximum bid price for a running instance, the instance is automatically termi-

^{a)} Electronic mail: christopher.poole@qut.edu.au

Type	API Name	Compute Units	Processors	RAM (GB)	Storage (GB)
Standard Small	<code>m1.small</code>	1	1 virtual	1.7	160
Standard Large	<code>m1.large</code>	4	4 virtual	7.5	850
Standard 1X Large	<code>m1.xlarge</code>	8	8 virtual	15	1690
Micro	<code>t1.micro</code>	2 (burst)	2 virtual	0.613	EBS
High Mem. 1X Large	<code>m2.xlarge</code>	6.5	2 virtual	17.1	420
High Mem. 2X Large	<code>m2.2xlarge</code>	13	4 virtual	34.2	850
High Mem. 4X Large	<code>m2.4xlarge</code>	26	8 virtual	68.4	1690
High CPU Medium	<code>c1.medium</code>	5	2 virtual	1.7	350
High CPU 1X Large	<code>c1.xlarge</code>	20	8 virtual	7	1690
Cluster 4X Large	<code>cc1.4xlarge</code>	33.5	2 Intel Xeon X5570	23	1690

TABLE I: Types of preconfigured instance types available to the user on EC2¹². The compute capability of the Micro type is burst only; the maximum compute cannot be sustained for lengthy periods. CPU bases instances are shown only; the Cluster GPU Quad Extra Large instance type is also available based on the Cluster 4X Large type with the addition of 2 NVIDIA Tesla Fermi M2050 GPU's¹².

nated. Further, hourly rates are not prorated for partial instance hour usage; the hourly runtime of each instance is rounded up to the nearest hour. Twenty instances running for half an hour each (10 hours of use) would be billed as 20 instance hours, whereas one instance running for 10 hours would be billed as only 10 instance hours for example.

Here within we describe in the process of executing a pre-existing GEANT4 simulation of a clinical linear accelerator¹⁶ on the Amazon EC2 computing resource. With a Python (Python Software Foundation, USA)¹⁷ interface to the simulation, the `boto` Python module for AWS is used to distribute jobs in the cloud environment from the local user machine.

II. METHODS

A. Clinical Linear Accelerator Simulation

A Varian Clinac was commissioned and calibrated for absolute dose calculation as described elsewhere¹⁶. A multi-step commissioning approach was used to tune the simulation so as to match depth dose and beam profile measurements in a water tank; the commissioning was carried out for a range of jaw defined field sizes using local compute resources. Further, the widely used intensity modulated radiation therapy (IMRT) verification test known as the chair test was simulated locally and compared to measurement¹⁶. All simulation calculations were verified with measurement using gamma evaluation and a pass/fail distance to agreement criterion of 3%/3 mm¹⁶.

Using the `boost::python` C++ libraries¹⁸ and the `g4py`¹⁹ Python bindings already present in the GEANT4 toolkit, an interface to the linear accelerator simulation was created; Python interface examples distributed with the GEANT4 toolkit served as a template. Instantiating the Python class `Linac` provided a basic linac set-up

with default values for parameters such as MLC and jaw positions and gantry rotation. Property constructs with both get and set methods such as `Linac.energy` allowed for direct access to all parameters that define linac operation and simulation configuration. Phantom geometry definition was also possible through the Python interface with `g4py` using standard techniques.

B. AWS Instance Set-up

A single instance of type `t1.micro` was launched using the pre-built and official Ubuntu 10.04 LTS 64 bit Amazon Machine Image (AMI) with identifier `ami-3202f25b`, booting from EBS. Elastic Block Storage was selected over the standard instance storage as EBS enables faster boot and persistence of data saved to disk after instance shutdown; however it should be noted that data saved to the instance disk would be lost on termination - distinct from shutdown as termination effectively destroys the instance¹³. The boot process itself was similar to the normal boot process for a default install of any recent version of the Ubuntu server distribution²⁰. Unlike a conventional local install however, the `libcloud`²¹ package was installed by default on the AMI enabling access to instance user data parsed to the instance at the time of creation. Using a public/private key-pair generated using the AWS dashboard, remote access and administration of the instance was established using a secure shell (SSH); the fully qualified Dynamic Name Server (DNS) address of the instance was made available to the user through the AWS dashboard (right click on instance → *Connect* menu item). GEANT4 version 9.3 and its dependencies were compiled and installed on the instance as well as other packages including `boost::python` and the `numpy` Numerical Python module²². Where available, pre-built binaries in the Ubuntu software repositories were favoured over compiling software from source. Once configured, the instance was saved as a custom and private AMI us-

ing the menu options available in the AWS dashboard (right click on instance → *Save instance as AMI* menu item)- this custom AMI was then available to boot up to 20 instances with the default AWS account set-up. In the case of booting 20 High CPU Extra Large EC2 instances, 160 CPU cores were made available to the user with a total compute capability of 400 EC2 units.

C. Distributing Jobs in the Cloud

Using `boto`, the Python API for AWS including EC2 and S3, a job launcher was created that managed the packing of a job description and data into a compressed archive and the launching of a group instances, see figure 1. For a given job, the simulation configuration included a manifest of all files and folders to be included as job data. Using the `tarfile` Python module, part of the Python standard library¹⁷, each file or folder in the manifest was added to an archive, followed by compression and writing to disk. From the local user machine, the compressed job archive was uploaded to S3 one time per unique simulation using `boto`. An EC2 reservation was requested which launched the prescribed number of instances for the job; a process fully managed by the `boto` Python module and EC2. Each instance had user data containing the simulation configuration including the location of the job archive on S3 transmitted to it automatically.

At instance boot time, a Python script was automatically executed, recovering the simulation configuration from the pre-transmitted user data and launching a pool of worker processes with a pool size equal to the number of processor cores available on the instance, see figure 2. The worker pool was created using the `multiprocessing` Python module¹⁷, again part of the Python standard library enabling a simulation described in Python function to be executed multiple times and concurrently across a number of processes equal to the pool size. On each instance, the master process managing the pool of worker processes waited for all workers to finish execution, subsequently combining and compressing the results returned by each worker process.

Finally, the compressed result was uploaded to S3 to a location specified in the simulation configuration and the instance was terminated as soon as possible, thus minimising the potential of cost escalation. Retrieving results from S3 could be performed using the AWS dashboard and a web-browser. For execution of instances on the spot market, a maximum bid price could be specified at the time of reservation and configured as a parameter along with all other simulation parameters. From the user perspective, there was no difference between an instance acquired on-demand or bid for on the spot market.

D. Benchmarking Performance and Cost

High CPU Extra Large EC2 instances were chosen for all jobs executed in the cloud as they provided the highest on-demand compute density per dollar, see section III C. A series of test simulations were performed so as to examine simulation performance as a function of EC2 instance count. Using the GEANT4 geometry primitive `G4Box`, a 40 cm cubic water phantom was defined and positioned with its center at the iso-center of the linear accelerator; 100 cm source to axis distance (SAD) or 80 cm source to surface distance (SSD). Irradiated with a jaw defined 5×5 cm field with gantry and primary collimator angles set to zero, 2.5×10^6 electrons incident on the copper target in the linear accelerator treatment head were simulated. The simulation was repeated for a range of EC2 instance counts ($1 \leq n \leq 20$) on the spot market (max price = 0.30 USD) with simulation completion time (the time elapsed from starting a job to uploading a result to S3), instance uptime, total simulation time (the total real CPU time used) and total simulation cost recorded. On-demand instance cost was calculated from the billed instance hours multiplied by the on-demand rate for the High CPU Extra Large instance type and compared to the actual cost incurred as a result of simulating the above using instances bid for on the spot market. Finally, historical data from January 1st to April 18th 2011 was acquired for each instance type using functionality provided by `boto` allowing for spot price history to be downloaded, and basic descriptive statistics were calculated.

III. RESULTS

A. Simulation Output

Figure 3 shows typical output for the simulation described in section IID using a 2 mm scoring dose grid. All dose values are shown normalised to the maximum central axis dose. The size in memory for the entire dose grid with $128 \times 128 \times 128$ voxels using single precision floating point values was 8 MB per worker process for a total of 64 MB per instance.

B. Compute Performance

For the simulation described in section IID the average time from instance boot to the start of the simulation on the same node was 59 ± 1 s. Figure 4(a) shows the simulation completion time t_c as a function of instance count; it was found to follow

$$t_c = \frac{t_s}{n_i n_p}, \quad (1)$$

where t_s is the total simulation time required, $n_i \in \mathbb{N}^* = \{1, 2, 3, \dots, 20\}$ is the number of instances used per job

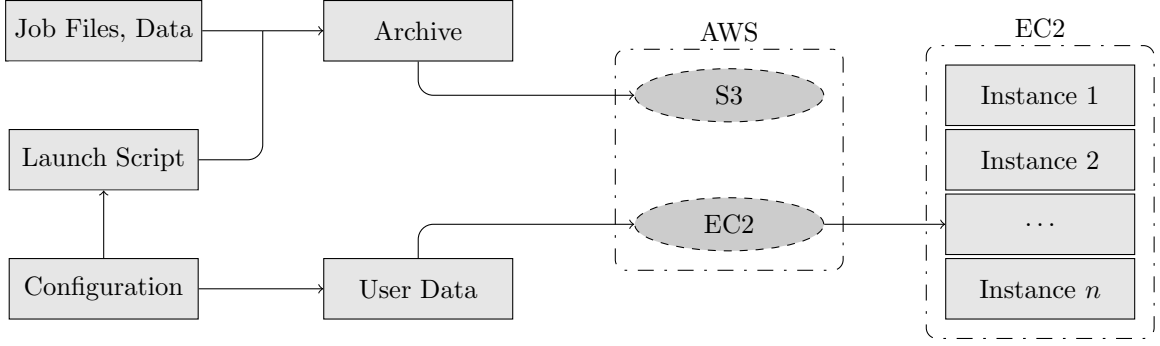


FIG. 1: Launching EC2 instances from the local user machine.

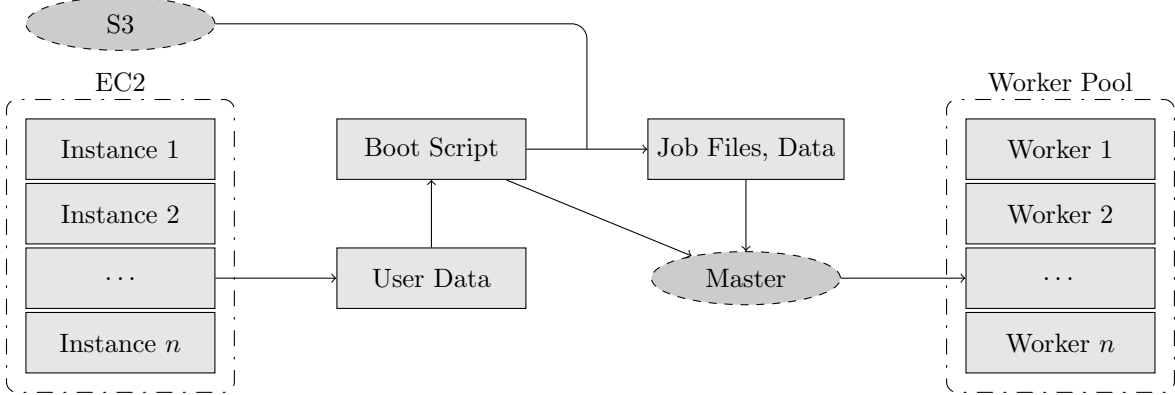


FIG. 2: Simulation configuration and worker pool creation on each EC2 instance.

and $n_p \in \mathbb{N}^* = \{1, 2, 3, \dots, 8\}$ is the number of processors available per instance. Noting that the default AWS accounts allowed for a maximum of $n_i = 20$ instances, and the maximum number of processors available per instances was $n_p = 8$ as of April 2011¹². Total simulation time or the total real CPU time consumed for the simulation as a function of instance count is shown in figure 4(b). Mean total simulation time required for the simulation described in section IID was $t_s = 26.1 \pm 0.2$ hours where the uncertainty represents one standard deviation about the mean.

C. Usage Costs

Historical spot prices for the year 2011 to April 18th for an Amazon EC2 High CPU Extra Large instance were acquired. A mean spot price of 0.34 ± 0.13 USD over this period was one half of the on-demand instance price (0.68 USD/hour) - a general trend observed for most EC2 instance types, see table I. At the time of simulation, the quoted spot price for an Amazon EC2 High CPU Extra Large instance was 0.223 USD/hour, approximately one third of the on-demand instance. Where the instance count was greater than the simulation completion time in

hours, cost escalation was linear with increasing instance count, see figure 5. Billable instances hours required to complete a given job requiring t_s total compute hours were found to follow

$$t_i = n_i \left\lceil \frac{t_s}{n_i n_p} \right\rceil = n_i \lceil t_c \rceil, \quad (2)$$

where $t_i \in \mathbb{N}^* = \{1, 2, 3, \dots\}$ is the total billable instance hours and $\lceil \dots \rceil$ indicates the ceiling function, noting that the uptime of a given instance was rounded up to the nearest hour for the purposes of billing. Simulations running at least total cost were found where the simulation time in hours was wholly divisible by the total number of instances running for that job, corresponding to the factors of $\lceil t_s/n_p \rceil \in \mathbb{N}^* = \{1, 2, 3, \dots\}$.

IV. DISCUSSION & CONCLUSION

Using a GEANT4 simulation of a clinical linear accelerator, executed on the Amazon Elastic Compute Cloud, we have demonstrated the potential usefulness of cloud computing for rapid radiotherapy dose calculation. Additionally, a simple formulation allowing for the optimal selection of instance count for least cost has been proposed,

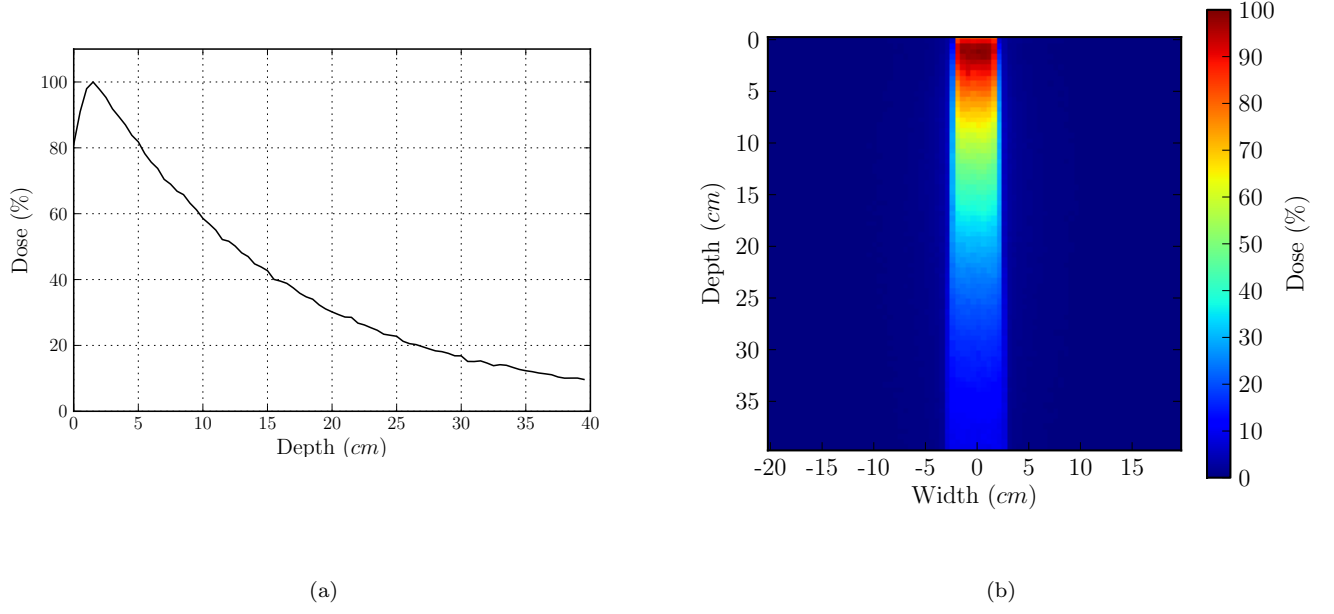


FIG. 3: Simulation output; (a) shows the central axis depth dose and (b) shows the dose distribution of the central slice in the water phantom. Note that the iso-center of the simulated linear accelerator was positioned at (0, 20) in (b).

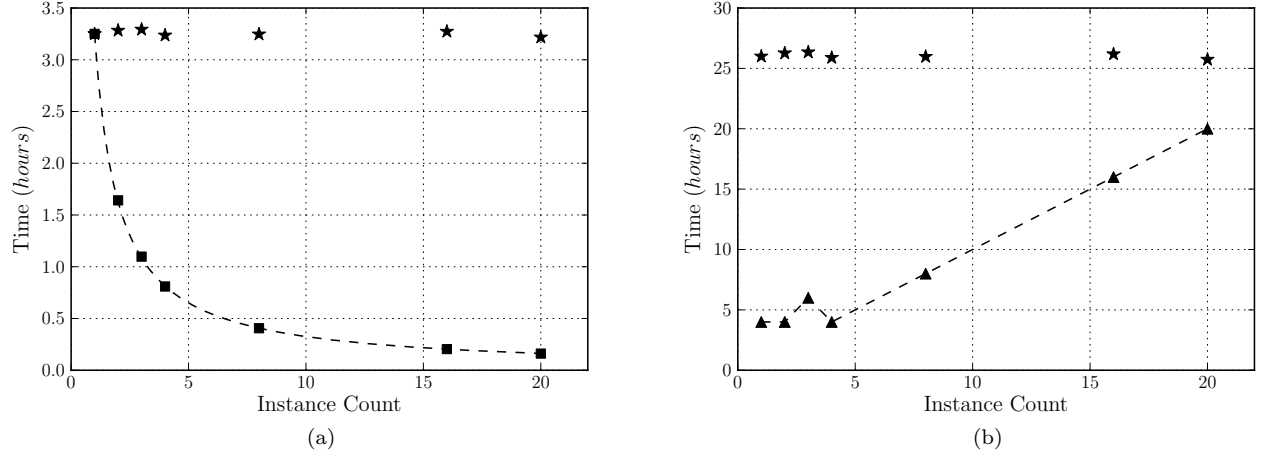


FIG. 4: Simulation time (a) where ★'s indicate the total instance up-time, ■'s indicate the time to simulation completion and the dashed line indicates the predicted simulation completion time (equation 1). Billable instance time (b) as a function of instance count where ★'s indicate the total compute required, ▲'s indicate the billable instance time, and the dashed line indicates the predicted billable instance time (equation 2).

given some estimate of total simulation time required. Figure 4(a) shows simulation time decreasing as $1/n$ with increasing instance count as observed by others⁸, cost however increases linearly with increasing instance count when simulation time in hours is less than the instance

count, as shown in figure 4(b). For a given simulation, if time is not a critical factor, the number of instances used can be tuned for least cost by ensuring each instance is in use for whole hours, as Amazon EC2 instances charges are not prorated for partial instance hour usage. How-

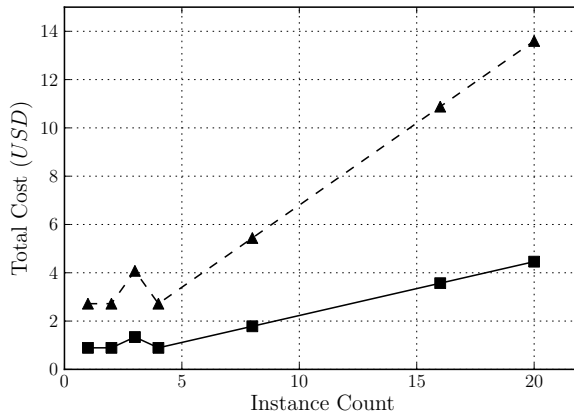


FIG. 5: Simulation cost as a function of instance count where ■'s indicate the incurred cost as a result of bidding for Amazon EC2 High CPU Extra Large instances on the spot market (0.223 USD/hour), ▲'s indicate the equivalent cost had the on-demand rate of 0.68 USD/hour been charged, and the solid and dashed lines indicate the predicted instances hours (equation 2) multiplied by the hourly rate.

API Name	Compute Units	Cost (USD/hr)			Cost ($\text{USD/hr/compute unit}$)	
		On-demand	Spot Average	diff %	On-demand	Spot Average
m1.small	1	0.085	0.043 ± 0.009	51	0.085	0.043
m1.large	4	0.34	0.16 ± 0.05	47	0.085	0.04
m1.xlarge	8	0.68	0.30 ± 0.09	45	0.085	0.038
t1.micro	2 (burst)	0.02	0.012 ± 0.003	60	0.01	0.006
m2.xlarge	6.5	0.50	0.21 ± 0.07	41	0.077	0.032
m2.2xlarge	13	1.00	0.44 ± 0.09	44	0.077	0.034
m2.4xlarge	26	2.00	0.87 ± 0.17	44	0.077	0.034
c1.medium	5	0.17	0.087 ± 0.03	51	0.034	0.017
c1.xlarge	20	0.68	0.34 ± 0.13	50	0.034	0.017
cc1.4xlarge	33.5	1.60	0.57 ± 0.06	35	0.048	0.017

TABLE II: Long-term spot market instance prices for the year 2011 to April 18th for a range of preconfigured EC2 instance types. Uncertainty in the average spot price is one standard deviation about the mean.

ever, in an environment where time is critical, increasing instance count reduces simulation time with a linearly increasing cost penalty.

Two fee structures were examined when considering EC2 usage; a direct comparison between the actual cost incurred as a result of using instances bid for on the spot market and the projected cost of acquiring the same instances had on-demand rates been charged. At the time of simulation, the spot market price of a single instance of type High CPU Extra Large was 0.223 USD/hour and approximately one third of the on-demand price for the same instance type. This was less than the long term average of $0.34 \pm 0.13 \text{ USD/hour}$ at one half of the on-demand price; a general trend observed for all instance types. Whilst volatility in the instance market may result in somewhat unpredictable expenditure, generally it is at least 50% cheaper to use the instance spot market to acquire EC2 instances for computation.

Application of this technique enables a GEANT4 user

to perform a simulation in a distributed compute environment, with a low entry cost and no express need for dedicated compute hardware. For clinics in developing countries for example, which may not have sufficient resources to provide adequate cancer care²³ much less manage dedicated compute hardware, this may be of particular benefit. Indeed, the shortfall in the quality of cancer care in developing countries has been identified by others^{23,24}, in particular the relationship between inadequate staff training and suboptimal treatment delivery²⁴. Systems to remedy this have been proposed by others, and of particular note is the Hospital Platform for E-health (HOPE)²⁵ enabling the remote verification of radiotherapy treatment plans and other diagnostic and therapeutic tests. Adoption of initiatives such as HOPE, coupled with the computational resources provided by the cloud and the simulation techniques described here within may offer significant scientific and social benefit.

Further work will explore any potential differences

in dose calculations performed using local computing resources and resources that are provided by the cloud. Presently this work is part of a software toolkit using GEANT4 for the simulation of clinical linear accelerators¹⁶. Source code for running GEANT4 simulations on EC2 as described here within is freely available and may be obtained from: <http://code.google.com/p/manysim/>

ACKNOWLEDGMENTS

This work is funded by the Queensland Cancer Physics Collaborative, and Cancer Australia (Department of Health and Ageing) Research Grant 614217.

- ¹S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, *et al.*, "Geant4 - a simulation toolkit," *Nuclear Instruments and Methods in Physics Research-Section A* Only **506**, 250–303 (2003).
- ²B. Caccia, C. Andenna, and G. A. P. Cirrone, "MedLinac2: a GEANT4 based software package for radiotherapy," *Annali dell'Istituto superiore di sanità* **46**, 173–177 (2010).
- ³S. Jan, D. Benoit, E. Becheva, T. Carlier, F. Cassol, P. Descourt, T. Frisson, L. Grevillot, L. Guigues, L. Maigne, *et al.*, "GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy," *Physics in Medicine and Biology* **56**, 881 (2011).
- ⁴E. Spezi and G. Lewis, "An overview of Monte Carlo treatment planning for radiotherapy," *Radiation protection dosimetry* (2008).
- ⁵L. Grevillot, T. Frisson, D. Maneval, N. Zahra, J. N. Badel, and D. Sarrut, "Simulation of a 6 MV Elekta Precise Linac photon beam using GATE/GEANT4," *Physics in Medicine and Biology* **56**, 903 (2011).
- ⁶P. Rodrigues, A. Trindade, L. Peralta, C. Alves, A. Chaves, and M. C. Lopes, "Application of GEANT4 radiation transport toolkit to dose calculations in anthropomorphic phantoms," *Applied Radiation and Isotopes* **61**, 1451–1461 (2004).
- ⁷J. Allison, K. Amako, J. Apostolakis, H. Araujo, P.A. Dubois, M. Asai, G. Barrand, R. Capra, S. Chauvie, R. Chytrcek, *et al.*, "Geant4 developments and applications," *Nuclear Science, IEEE Transactions on* **53**, 270–278 (2006).
- ⁸R.W. Keyes, C. Romano, D. Arnold, and S. Luan, "Radiation therapy calculations using an on-demand virtual cluster via cloud computing," *Arxiv preprint arXiv:1009.5282* (2010).
- ⁹J. Gruntorad and M. Lokajicek, "International Conference on Computing in High Energy and Nuclear Physics (CHEP'09)," in *Journal of Physics: Conference Series*, Vol. 219 (2010) p. 001001.
- ¹⁰A. Farbin, "Emerging Computing Technologies in High Energy Physics," *Arxiv preprint arXiv:0910.3440* (2009).
- ¹¹A Silverman, I Fedorko, W Lapka, and G Lo Presti, "CHEP 2010 Report. CHEP - Computing in High Energy and nuclear Physics," *Tech. Rep. CERN-IT-Note-2010-007* (CERN, Geneva, 2010).
- ¹²"Amazon EC2 Instance Types," Amazon Web Services LLC. <http://aws.amazon.com/ec2/instance-types/> (April 2011).
- ¹³"Amazon EBS," Amazon Web Services LLC. <http://aws.amazon.com/ebs/> (April 2011).
- ¹⁴M. Garnaat *et al.*, *Boto Python interface to Amazon Web Services Documentation*, Computer software. <http://code.google.com/p/boto/>, v2.0 ed. (2010).
- ¹⁵"Amazon EC2 Pricing," Amazon Web Services LLC. <http://aws.amazon.com/ec2/pricing/> (April 2011).
- ¹⁶I. Cornelius, B. Hill, N. Middlebrook, C. Poole, B. Oborn, and C. Langton, "Commissioning of a Geant4 based treatment plan simulation tool: linac model and DICOM-RT interface," *Arxiv preprint arXiv:1104.5082* (Apr. 2011).
- ¹⁷G. van Rossum and F. L. Drake, *Python Reference Manual*, Python Software Foundation. <http://python.org/>, v2.7.1 ed. (April 2011).
- ¹⁸D. Abrahams, U. Koethe, RW Grosse-Kunstleve, *et al.*, *The Boost Python Library Documentation*, Computer software. <http://boost.org/libs/python/>, v1.41 ed. (October 2004).
- ¹⁹K. Murakami and H. Yoshida, "A Geant4-Python Interface: Development and Its Applications," in *Nuclear Science Symposium Conference Record, 2006. IEEE*, Vol. 1 (IEEE, 2006) pp. 98–100.
- ²⁰Ubuntu Community Documentation. <https://help.ubuntu.com/community/EC2StartersGuide>, *Ubuntu EC2 Starters Guide* (March 2011).
- ²¹Computer software. <http://ci.apache.org/projects/libcloud/apidocs/>, *Libcloud: a unified interface to the cloud*, v0.4.2 ed. (2011).
- ²²D. Ascher, P.F. Dubois, K. Hinsén, J. Hugunin, T. Oliphant, *et al.*, *Numerical Python Documentation*, Computer software. <http://numpy.scipy.org/>, v1.5 ed.
- ²³T.P. Hanna and A.C.T. Kangolle, "Cancer control in developing countries: using health data and health services research to measure and improve access, quality and efficiency," *BMC International Health and Human Rights* **10**, 24 (2010), ISSN 1472-698X.
- ²⁴T.P. Shakespeare, M.F. Back, J.J. Lu, K.M. Lee, and R.K. Mukherjee, "External audit of clinical practice and medical decision making in a new Asian oncology center: results and implications for both developing and developed nations," *International Journal of Radiation Oncology* Biology* Physics* **64**, 941–947 (2006).
- ²⁵M. Diarena, S. Nowak, JY Boire, V. Bloch, D. Donnarieix, A. Fessy, B. Grenier, B. Irrthum, Y. Legré, L. Maigne, *et al.*, "HOPE, an open platform for medical data management on the grid.." *Studies in health technology and informatics* **138**, 34 (2008).